Flex static code analysis

Unique rules to find Bugs, Security Hotspots, and Code Smells in your FLEX code

Secrets
ABAP
Apex
C
C++
CloudFormation
COBOL
C#
CSS
Flex
Go
HTML
Java
JavaScript
Kotlin
Kubernetes
Objective C
PHP
PL/I
PL/SQL
Python
RPG
Ruby
Scala
Swift
Terraform
Text
TypeScript
T-SQL
VB.NET
VB6
XML

All rules 76 | 🔒 Vulnerability 5 | 🐛 Bug 9 | 🛡 Security Hotspot 1 | ☢ Code Smell 61

Tags ⌄ | Search by name...

Security.allowDomain(...) should only be used in a tightly focused manner
🔒 Vulnerability

The flash.system.Security.exactSettings property should never be set to false
🔒 Vulnerability

Dynamic classes should not be used
☢ Code Smell

"LocalConnection" should be configured to narrowly specify the domains with which local connections to other Flex application are allowed
🔒 Vulnerability

"default" clauses should be first or last
☢ Code Smell

Event types should be defined in metadata tags
☢ Code Smell

Event names should not be hardcoded in event listeners
☢ Code Smell

The special "star" type should not be used
☢ Code Smell

Variables of the "Object" type should not be used
☢ Code Smell

Methods should not be empty
☢ Code Smell

Constant names should comply with a naming convention
☢ Code Smell

All branches in a conditional structure should not have exactly the same implementation
🐛 Bug

Classes that extend "Event" should

Public classes, methods, properties and metadata should be documented with ASDoc

Analyze your code

☢ Code Smell   🔴 Major ❓   🏷 convention

Try to imagine using the standard Flex API without ASDoc. It would be a nightmare, because ASDoc is the only way to understand of the contract of the API.

Documenting an API with ASDoc increases the productivity of the developers use it.

Noncompliant Code Example

```
public class MyClass {
    public var myLabel:String;

    public function myMethod(param1:String):Boolean {...}
}
```

Compliant Solution

```
/**
 * my doc
 */
public class MyClass {
    /**
     * my doc
     */
    public var myLabel:String;

    /**
     * my doc
     * @param param1 my doc
     * @return my doc
     */
    public function myMethod(param1:String):Boolean {...}
}
```

Exceptions

Classes or class elements with an ASDoc @private comment are ignored by this rule.

```
/**
 * @private  // This class and all its elements are igno
 */
public class MyClass {  // Compliant

    public var myLabel:String;   // Compliant
}

public class AnotherClass {  // Noncompliant; class not

    /**
     * @private
     */
```

```
    public var name:String;   // Compliant
}
```

override "Event.clone()"
🐞 Bug

Constructors should not dispatch events
🐞 Bug

"ManagedEvents" tags should have companion "Event" tags
🐞 Bug

Objects should not be instantiated inside a loop
☢ Code Smell

Two branches in a conditional structure should not have exactly the same implementation
☢ Code Smell

Constructor bodies should be as lightweight as possible
☢ Code Smell

Only "while", "do" and "for" statements should be labelled
☢ Code Smell

Statements, operators and keywords specific to ActionScript 2 should not be used
☢ Code Smell

"for" loop stop conditions should be invariant
☢ Code Smell

Unused function parameters should be removed
☢ Code Smell